HITACHI
Inspire the Next

Hitachi Hyper Scale-Out Platform (HSP)

Cassandra VM Deployment Guide

Hitachi Data Systems

MK-95HSP027-00
06 October 2016

Contents

Preface

This document describes and provides instructions for deploying the Cassandra template on Hitachi Hyper Scale-Out Platform (HSP).

This Preface includes the following information:

Intended audience

This document is intended for system administrators, Hitachi Data Systems representatives, and authorized service providers who need to deploy the Cassandra VM template on Hitachi Hyper Scale-Out Platform (HSP).

Readers of this document should be familiar with the following:

- Cassandra administration
- Linux operating system and working in a restricted shell environment
- Site-specific network information

Product version

This document applies to Hyper Scale-Out Platform release 1.2.037 or later.

Release notes

The release notes for this product are available on Hitachi Data Systems Support Connect: https://support.hds.com/en_us/contact-us.html. Read the release notes before installing and using this product. They may contain requirements or restrictions that are not fully described in this document or updates or corrections to this document.

Document revision level

Revision	Date	Description
MK-95HSP027-00	06 October 2016	Initial release

Document conventions

This document uses the following typographic conventions:

Convention	Description
<Italic> in angle brackets	Indicates a variable, which is a placeholder for site- or installation-specific details that you need to provide. For example: `copy <source-file> <target-file>`
`monospace`	Indicates text that is displayed on the screen or text that you need to enter. For example: `# pairdisplay -g oradb` Also, the name of a directory, folder, or file. For example: The `horcm.conf` file...

This document uses the following icons to draw attention to information:

Icon	Meaning	Description
	Tip	Provides helpful information, guidelines, or suggestions for performing tasks more effectively.
	Important	Calls attention to information that is essential to the completion of a task.
	Caution	Warns that failure to take or avoid a specified action could result in adverse conditions or consequences (for example, loss of access to data).
	Warning	Warns that failure to take or avoid a specified action could result in severe conditions or consequences (for example, loss of data).

Conventions for storage capacity

Hyper Scale-Out Platform uses the International Electrotechnical Commission (IEC) binary definition for storage capacity units:

Capacity unit	Value
1 kibibyte (KiB)	1,024 (2^{10}) bytes
1 mebibyte (MiB)	$1,024^2$ (2^{20}) bytes
1 gibibyte (GiB)	$1,024^3$ (2^{30}) bytes
1 tebibyte (TiB)	$1,024^4$ (2^{40}) bytes
1 pebibyte (PiB)	$1,024^5$ (2^{50}) bytes
1 exbibyte (EiB)	$1,024^6$ (2^{60}) bytes

Accessing product documentation

Product documentation is available on Hitachi Data Systems Support Connect: https://knowledge.hds.com/Documents. Check this site for the most current documentation, including important updates that may have been made after the release of the product.

Getting help

Hitachi Data Systems Support Portal is the destination for technical support of products and solutions sold by Hitachi Data Systems. To contact technical support, log on to Hitachi Data Systems Support Connect for contact information: https://support.hds.com/en_us/contact-us.html.

Hitachi Data Systems Community is a global online community for HDS customers, partners, independent software vendors, employees, and prospects. It is the destination to get answers, discover insights, and make connections. **Join the conversation today!** Go to community.hds.com, register, and complete your profile.

Comments

Please send us your comments on this document:

hsp.documentation.comments@hds.com

Include the document title and part number, including the revision (for example, -01), and refer to specific sections and paragraphs whenever possible. All comments become the property of Hitachi Data Systems.

Thank you!

1

Fitting the SSDs

This chapter describes the procedure for replacing some of the hard drives in the HSP cluster with SSDs so they can be used by Cassandra.

Note:

- Cassandra is very sensitive to latency so it is recommended that data files and commit logs reside on SSDs.
- It is recommended that the SSD drives are fitted to the cluster nodes as soon as possible, ideally when the cluster is fresh from the factory. This is because the process takes some time to complete on a new cluster and the time taken increases with the amount of data that is written on the disks.

If your HSP cluster was delivered fitted with SSDs this section can be ignored, move on to the Preparing the VM ISO and RPMs for installation section.

Before you begin

Verify that you have:

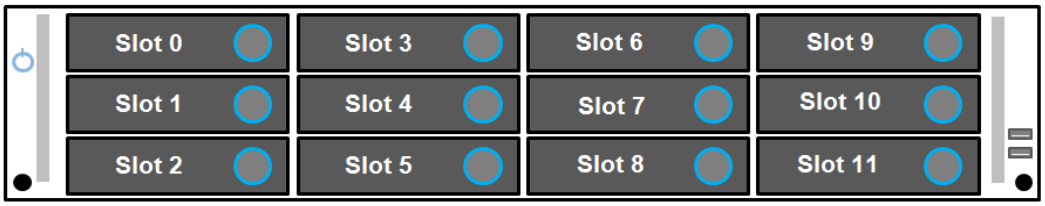
- An initialized HSP cluster running at HSP version 1.2.037 or later. Ensure that your HSP cluster has its DNS domain name, DNS server and NTP server configured correctly.
- The SSDs supplied by the factory for installation into the HSP nodes.

Procedure

Preparation

If your HSP cluster was delivered fitted with SSDs this section can be ignored, move on to the Preparing the VM ISO and RPMs for installation

1. Identify the hard disk drives you want to replace but do not replace any drives until instructed to do so.



Tip: To make hardware and software administration easier replace drives in the same drive slots for each node of the cluster.

For Example:
If you are replacing two drives in each node of a cluster replace the drives in **Slot 0** and **Slot 1** of each node.

This is not a requirement but using different drive slots in each node complicates configuration and future administration.

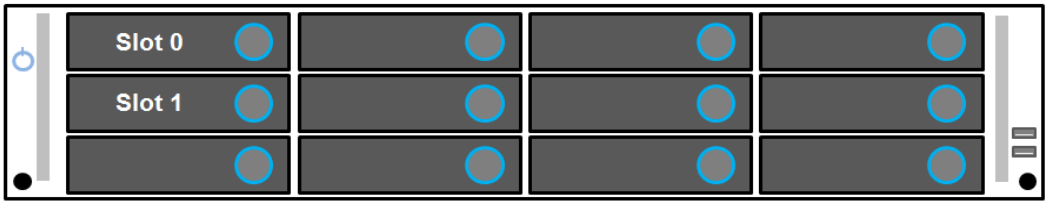
2. SSH into the cluster virtual IP and list the disks in Node 001:

```
admin@Node001:~$ hspadm disk list --node Node001
```

Example output:

```
-------------------------------------------------------------------------------------------------------------
| Name         | Run State | Role    | Identify | Used Capacity | Capacity    | Serial Number          | Media Type |
-------------------------------------------------------------------------------------------------------------
| Node001_S0   | UP        | cluster | No       | 4.993 GiB     | 5.457 TiB   | S4D0FQPY0000K614E7RD | HDD        |
| Node001_S1   | UP        | cluster | No       | 6.909 GiB     | 5.457 TiB   | S4D0EWWA0000K6145VAP | HDD        |
| Node001_S10  | UP        | cluster | No       | 5.224 GiB     | 5.457 TiB   | S4D0EY0P0000K612FMDW | HDD        |
| Node001_S11  | UP        | cluster | No       | 6.551 GiB     | 5.457 TiB   | S4D0FSE40000K61509S5 | HDD        |
| Node001_S2   | UP        | cluster | No       | 6.092 GiB     | 5.457 TiB   | S4D0FQEM0000K6150WX1 | HDD        |
| Node001_S3   | UP        | cluster | No       | 5.152 GiB     | 5.457 TiB   | S4D0FRW30000K6150WRJ | HDD        |
| Node001_S4   | UP        | cluster | No       | 4.122 GiB     | 5.457 TiB   | S4D0FSJZ0000K6150X7Q | HDD        |
| Node001_S5   | UP        | cluster | No       | 6.014 GiB     | 5.457 TiB   | S4D0FSAS0000K6150XW3 | HDD        |
-------------------------------------------------------------------------------------------------------------
```

In this example the hard drives in `Node001_S0` and `Node001_S1` will be replaced. These are the two drives shown on the diagram below:



Note: If you have created vm-volumes on the disks you are replacing delete them before proceeding.

3. Change the role of the drives to be `vm` instead of `cluster`:

```
admin@Node001:~$ hspadm disk edit --node Node001 --name Node001_S0 --role vm
admin@Node001:~$ hspadm disk edit --node Node001 --name Node001_S1 --role vm
```

Each of the commands above can take some time to complete as data is moved off the drive. On a heavily used cluster this can take hours.

Repeat the above commands for all drives that need to be replaced in the cluster.

In this example, these would be drives `Node002_S0`, `Node002_S1`, `Node003_S0`, etc.

4. Check the drive role status and wait for the role of both drives to change to VM before moving to the next step.

```
admin@Node001:~$ hspadm disk list --node Node001
```

Example output:

```
-------------------------------------------------------------------------------------------------------------------
| Name        | Run State | Role    | Identify | Used Capacity | Capacity    | Serial Number        | Media Type |
-------------------------------------------------------------------------------------------------------------------
| Node001_S0  | UP        | vm      | No       | 4.993 GiB     | 5.457 TiB   | S4D0FQPY0000K614E7RD | HDD        |
| Node001_S1  | UP        | vm      | No       | 6.909 GiB     | 5.457 TiB   | S4D0EWWA0000K6145VAP | HDD        |
| Node001_S10 | UP        | cluster | No       | 5.224 GiB     | 5.457 TiB   | S4D0EY0P0000K612FMDW | HDD        |
| Node001_S11 | UP        | cluster | No       | 6.551 GiB     | 5.457 TiB   | S4D0FSE40000K61509S5 | HDD        |
| Node001_S2  | UP        | cluster | No       | 6.092 GiB     | 5.457 TiB   | S4D0FQEM0000K6150WX1 | HDD        |
| Node001_S3  | UP        | cluster | No       | 5.152 GiB     | 5.457 TiB   | S4D0FRW30000K6150WRJ | HDD        |
| Node001_S4  | UP        | cluster | No       | 4.122 GiB     | 5.457 TiB   | S4D0FSJZ0000K6150X7Q | HDD        |
| Node001_S5  | UP        | cluster | No       | 6.014 GiB     | 5.457 TiB   | S4D0FSAS0000K6150XW3 | HDD        |
-------------------------------------------------------------------------------------------------------------------
```

5. Remove the hard drives being replaced with SSDs from the cluster with the following command.

```
admin@Node001:~$ hspadm disk delete --node Node001 --name Node001_S0
Node001_S0 will be deleted. Continue? (y/n)
admin@Node001:~$ hspadm disk delete --node Node001 --name Node001_S1
Node001_S1 will be deleted. Continue? (y/n)
```

   Repeat the above command for all drives that need to be removed from the cluster.

   In this example, these would be drives `Node002_S0`, `Node002_S1`, `Node003_S0`, etc.

6. When the command completes and the drive is not shown in the output from `hspadm disk list`, you are ready to install the SSDs.

Fitting the SSDs

1. Remove the drive from the drive slot.
2. Before fitting make sure the SSD is securely screwed into its carrier. Insert the SSD carrier assembly into the vacant drive slot.
3. Repeat for each drive being replaced with an SSD.
4. When the SSDs are accepted by the cluster, they will show up in output from `hspadm disk list`. This can take up to an hour.

   Example Output:

```
-------------------------------------------------------------------------------------------------------------
| Name        | Run State | Role    | Identify | Used Capacity | Capacity    | Serial Number         | Media Type |
-------------------------------------------------------------------------------------------------------------
| Node001_S0  | UP        | vm      | No       | 0.0 KiB       | 744.675 GiB | BTTV537303B6800JGN    | SSD        |
| Node001_S1  | UP        | vm      | No       | 0.0 KiB       | 744.675 GiB | BTTV5373035H800JGN    | SSD        |
| Node001_S10 | UP        | cluster | No       | 5.224 GiB     | 5.457 TiB   | S4D0EY0P0000K612FMDW  | HDD        |
| Node001_S11 | UP        | cluster | No       | 6.551 GiB     | 5.457 TiB   | S4D0FSE40000K61509S5  | HDD        |
| Node001_S2  | UP        | cluster | No       | 6.092 GiB     | 5.457 TiB   | S4D0FQEM0000K6150WX1  | HDD        |
| Node001_S3  | UP        | cluster | No       | 5.152 GiB     | 5.457 TiB   | S4D0FRW30000K6150WRJ  | HDD        |
| Node001_S4  | UP        | cluster | No       | 4.122 GiB     | 5.457 TiB   | S4D0FSJZ0000K6150X7Q  | HDD        |
| Node001_S5  | UP        | cluster | No       | 6.014 GiB     | 5.457 TiB   | S4D0FSAS0000K6150XW3  | HDD        |
-------------------------------------------------------------------------------------------------------------
```

2

Preparing the VM ISO and RPMs for installation

This chapter describes the procedure for moving the Cassandra VM ISO and RPMs to the HSP cluster. At the end of this procedure the VM will be ready to be attached to the SSDs.

Before you begin

Verify that you have:

- An initialized HSP cluster running at HSP version 1.2.037 or later. Ensure that your HSP cluster has its DNS domain name, DNS server and NTP server configured correctly.
- A downloaded copy of the CentOS7-vanilla VM template on an existing HSP share. This ISO is available from Technical Information Service Center (TISC), and may be called something like `CentOS-7.2-baseVM-HSP_r2.tgz`
- Linux client.

Procedure

First, add the vm-template for the CentOS7 base image to the HSP cluster.

1. Extract the qcow2 image from the tgz file on the Linux client:

   ```
   linux# tar xzvf CentOS-7.2-baseVM-HSP_r2.tgz
   ```

2. Mount a share from the HSP cluster on your Linux system.

   ```
   linux# mount <HSP Cluster Virtual IP Address>:/<myshare> /mnt/hsp
   ```
   Where `/myshare` is the name of the available share.

3. Copy the qcow2 image from the ISO to the share:

   ```
   linux# cp CentOS-7.2-baseVM-HSP_r2.qcow2 /mnt/hsp
   ```

4. On the HSP cluster, add the vm-template:

   ```
   admin@Node001:~$ hspadm vm-template add --name centos7-vanilla --
   image-path myshare:/CentOS-7.2-baseVM-HSP_r2.qcow2
   ```

5. Verify this was successful:

```
admin@Node001:~$ hspadm vm-template list
```

Example output

```
-------------------------------------------------------------------------------------
| Name            | Golden | Image Format | Creation Time       | Modification Time   |
-------------------------------------------------------------------------------------
| centos7-vanilla | No     | auto-detect  | 2016-09-05 18:49:17 | 2016-09-05 18:49:17 |
-------------------------------------------------------------------------------------
```

6. Now clone the CentOS7 vanilla template:

```
admin@Node001:~$ hspadm vm-template clone --name centos7-vanilla --
clone-name cassandra-template
```

7. Verify this was successful:

```
admin@Node001:~$ hspadm vm-template list
```

Example output

```
-------------------------------------------------------------------------------------
| Name              | Golden | Image Format | Creation Time       | Modification Time  |
-------------------------------------------------------------------------------------
| cassandra-template | No     | auto-detect  | 2016-09-05 18:52:54 | 2016-09-05 18:52:54|
| centos7-vanilla    | No     | auto-detect  | 2016-09-05 18:49:17 | 2016-09-05 18:49:17|
-------------------------------------------------------------------------------------
```

Note: From HSP version 1.2 it is no longer possible to directly start a VM template in sandbox mode. Instead, a VM instance is created in sandbox mode and changes are then made to the VM instance.

When the VM instance is deleted in sandbox mode any changes made are saved to the original template. Any VM instances created from this template will then include the changes made in sandbox mode.

8. Create a VM instance in sandbox mode using cassandra-template:

```
hspadm vm-instance add --name cass-sandbox --vm-template cassandra-
template --vm-size jumbo --is-sandbox Y
```

9. The CentOS7 template includes an SSH key for password-less login which is very useful for logging in between Cassandra nodes.

   You can either:

   - Use the default key which is included in the cass-sandbox VM.
     Go to Download RPMs.

   or

   - Change the ssh key
     Go to Change the ssh key (optional steps).

Change the ssh key (optional steps)

If you want to change the ssh key, please follow the steps in this section.

a. Obtain the IP address of the sandbox VM instance:

```
admin@Node001:~$ hspadm vm-instance list
```

Example output

```
-------------------------------------------------------------------------------------------------------------------
| Name         | Run State | Instance Group | IP Address        | VNC Address         | Memory     | CPUs | Sandbox |
-------------------------------------------------------------------------------------------------------------------
| cass-sandbox | UP        |                | 172.20.252.223/24 | 172.20.252.205:5900 | 16.000 GiB | 4    |  True   |
-------------------------------------------------------------------------------------------------------------------
```

b. Log into the sandbox using the IP address found in the previous step:

```
ssh 172.20.252.223
```

Username: root

Password: smrace1

c. Generate a new ssh key.

First, look at the contents of the existing ssh private key.

```
[root@cass-sandbox ~]# cat .ssh/id_rsa
```

Now create the new ssh key.

```
[root@cass-sandbox ~]# ssh-keygen
```

Output

```
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
…
```

The new ssh key is displayed.

d. Examine the contents of the new ssh private key to verify that it has changed.

```
[root@cass-sandbox ~]# cat .ssh/id_rsa
```

You can now use this new ssh key in your Cassandra installation.

e. Update authorized_keys:

```
[root@cass-sandbox ~]# cat .ssh/id_rsa.pub > .ssh/authorized_keys
```

f. Go to Download RPMs.

Download RPMs

If the HSP cluster is in a site where Internet access is disabled, then you may wish to download the RPMs to a gateway server and then copy them to cass-sandbox.

1. Cassandra requires Java. Please download the latest version of the JDK from:

   http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

   The JDK RPM is in the format `jdk-<version number>-linux-x64.rpm`. Since all Cassandra instances will require Java, move the RPM over to cass-sandbox and install it:

   ```
   [root@cass-sandbox ~]# rpm -ivh jdk-8u101-linux-x64.rpm
   ```

2. Now download the Cassandra RPMs. We will describe the procedure to download the community RPMs from Datastax, however if you install either DSE or code from cassandra.apache.org, the process is similar.

Note: Do not install Cassandra yet! We will need to prepare the disks for the data files and commit logs first.

3. Add the Apache Cassandra 3.0 repository to `/etc/yum.repos.d/datastax.repo`:

   ```
   [datastax]
   name = DataStax Repo for Apache Cassandra
   baseurl = http://rpm.datastax.com/community
   enabled = 1
   gpgcheck = 0
   ```

4. Download Cassandra RPMs to `/root/downloads`:

   ```
   [root@cass-sandbox ~]# mkdir -p /root/downloads
   [root@cass-sandbox ~]# yum install dsc30 cassandra30-tools -y --
   downloadonly --downloaddir=/root/downloads/
   ```

5. We now have all the software we need for a Cassandra VM instance. Delete the sandbox VM:

```
admin@Node001:~$ hspadm vm-instance delete --name cass-sandbox
```

OUTPUT

```
cass-sandbox will be deleted. Continue? (y/n)y
Deleting now...
admin@Node001:~$
```

As noted earlier, when the VM instance is deleted in sandbox mode any changes made are saved to the original template. Any VM instances created from the `cassandra-template` template will include the changes made in sandbox mode.

3

Direct attach drives to a VM

This chapter describes the procedure for attaching drives to Cassandra VMs. The example in this section describes a cluster with two SSDs per node but the processes can be applied to clusters with more or less SSDs. At the end of this procedure you will be ready to install Cassandra on your cluster.

Important: Because Cassandra is very sensitive to latency it is recommended that data files and commit logs reside on SSDs.

Procedure

Create VM-volumes

Note: The example in this section describes a cluster fitted with two SSDs per node. The same process is used if you have more or less SSDs in your cluster.

1. Create vm-volumes from the disks:

```
admin@Node001:~$ hspadm vm-volume add --name N10 --disk Node001_S0 --size 744GiB
admin@Node001:~$ hspadm vm-volume add --name N11 --disk Node001_S1 --size 744GiB
```

   In this example, the volume created is approximately the same size as the SSD. It is also possible to create multiple volumes on a disk. For example, if you want to run two VMs on an HSP node and have them share the SSD.

2. Repeat these commands for all disks in the cluster. You will need to assign at least one SSD volume to each Cassandra VM.

Create VM instances

1. Create VM instances for Cassandra:

   ```
   admin@Node001:~$ hspadm vm-instance add --name cass01 --vm-template
   cassandra-template --vm-size jumbo --use-address-pool Y --disk vm-
   volume-name=N10,vm-volume-name=N11
   ```

   In this example vm-size jumbo has been used, this is a standard VM size in HSP. You can also create a custom vm-size using the command `hspadm vm-size add`. A popular VM size for Cassandra VMs is 16GiB RAM and 6 cores.

2. Verify the volumes have been assigned to cass01:

   ```
   admin@Node001:~$ hspadm vm-volume list --name N10
   ```

   Example outputs

```
-------------------------------------------------------------------------------------
| Name | Run State | VM Instance | Disk       | Used Bytes | Free Bytes  | Total Bytes |
-------------------------------------------------------------------------------------
| N10  | UP        | cass01      | Node001_S0 | 0.0 KiB    | 744.000 GiB | 744.000 GiB |
-------------------------------------------------------------------------------------
```

```
admin@Node001:~$ hspadm vm-volume list --name N11
```

```
-------------------------------------------------------------------------------------
| Name | Run State | VM Instance | Disk       | Used Bytes | Free Bytes  | Total Bytes |
-------------------------------------------------------------------------------------
| N11  | UP        | cass01      | Node001_S1 | 0.0 KiB    | 744.000 GiB | 744.000 GiB |
-------------------------------------------------------------------------------------
```

3. Repeat this command for the number of Cassandra nodes you wish to create in your cluster.

Create RAID volumes

The steps in this section must be run on every VM that needs to be added to the Cassandra cluster.

1. Login to the VM and verify that the drives are available to the operating system:

```
[root@cass01 ~]# fdisk -l
```

Output

```
Disk /dev/vda: 107.4 GB, 107374182400 bytes, 209715200 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000123f5

   Device Boot      Start         End      Blocks   Id  System
/dev/vda1            2048   209715199   104856576   83  Linux

Disk /dev/vdb: 10 MB, 10485760 bytes, 20480 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/vdc: 798.9 GB, 798863917056 bytes, 1560281088 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/vdd: 798.9 GB, 798863917056 bytes, 1560281088 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

In the output above:

Root drive: `/dev/vda`

Shared drive: `/dev/vdb` (the drive that the VM gets its IP address from)

First SSD drive: `/dev/vdc`

Second SSD drive: `/dev/vdd` (if fitted)

Third SSD drive: `/dev/vde` (if fitted)

Tip: HDS recommend a RAID0 configuration using Linux software RAID for the SSDs attached to the VM. Other RAID configurations such as RAID1 and RAID5 are also possible. However, as Cassandra performs replication and failover itself RAID0 is recommended for performance.

2. The CentOS7-vanilla template that the cassandra-template is derived from includes RPMs for md (multiple device), which is Linux software RAID. Log into the VM as root, and run the following commands to configure RAID0:

   The following command assumes two RAID devices, the local SSD devices are `/dev/vdc` and `/dev/vdd`. `--level=stripe` indicates RAID0, `--chunk=4` indicates that the chunk size on your RAID device will be 4K.

   ```
   [root@cass01 ~]# mdadm --create --verbose /dev/md0 --level=stripe --
   raid-devices=2 --chunk=4 /dev/vdc /dev/vdd
   ```

3. Verify that RAID group was created with RAID0 and 4K chunk size:

   ```
   [root@cass01 ~]# mdadm --misc -D /dev/md0
   ```

   Output

   ```
   /dev/md0:
           Version : 1.2
     Creation Time : Thu Aug 18 21:40:31 2016
        Raid Level : raid0
        Array Size : 1560018944 (1487.75 GiB 1597.46 GB)
      Raid Devices : 2
     Total Devices : 2
       Persistence : Superblock is persistent

       Update Time : Thu Aug 18 21:40:31 2016
             State : clean
    Active Devices : 2
   Working Devices : 2
    Failed Devices : 0
     Spare Devices : 0

        Chunk Size : 4K

              Name : cass01:0  (local to host cass01)
              UUID : d5157f34:2ce84ecb:2bb69822:ce12dda8
            Events : 0

       Number   Major   Minor   RaidDevice State
          0     253       32        0      active sync   /dev/vdc
          1     253       48        1      active sync   /dev/vdd
   ```

   ```
   [root@cass01 ~]# cat /proc/mdstat
   ```

Output

```
Personalities : [raid0]
md0 : active raid0 vdc[0] vdd[1]
      1560018944 blocks super 1.2 4k chunks

unused devices: <none>
```

4. Add entry to `/etc/rc.local` to start md on boot.

```
[root@cass01 ~]# echo "mdadm --assemble --scan" >> /etc/rc.local
```

5. Create startup file for md:

```
[root@cass01 ~]# mdadm --detail --scan > /etc/mdadm.conf
```

6. Add entry in `/etc/fstab`

```
[root@cass01 ~]# echo "/dev/md0 /var/lib/cassandra ext4
discard,nobarrier 1 1" >> /etc/fstab
```

7. Create file-system for Cassandra and mount it.
   This final step assumes that the data files, commit log, hints, etc. are sub folders of `/var/lib/cassandra`

```
[root@cass01 ~]# mkdir -p /var/lib/cassandra
[root@cass01 ~]# mkfs -t ext4 /dev/md0
[root@cass01 ~]# mount /dev/md0
```

4

Installing and starting Cassandra

This chapter describes the procedure for installing and starting Cassandra VMs. At the end of this procedure Cassandra will be installed, running and ready for use.

- Before you begin
- Procedure

Before you begin

Verify that you have:

- Fitted the SSDs, see Fitting the SSDs.
- Prepared the VMs and RPMs for installation, see Preparing the VM ISO and RPMs for installation.
- Attached SSDs to the VMs, see Direct attach drives to a VM.
- The correct version of `cassandra.yaml` installed on the nodes. If you don't have one, contact the HSP apps team.

Procedure

Installing Cassandra

Install Cassandra using the following command:

```
[root@cass01 ~]# cd /root/downloads/
[root@cass01 downloads]# RPMS=`/bin/ls`
[root@cass01 downloads]# for r in $RPMS; do rpm -ivh $r; done
```

Update /etc/hosts

The `/etc/hosts` file must be updated on all of the VMs that are part of the Cassandra cluster.

1. Find the names and IP addresses of the Cassandra VMs using:

   ```
   admin@Node001:~$ hspadm vm-instance list
   ```

2. The `/etc/hosts` file on every VM must have new entries below the line `## end: analytics vm`

   Example

   ```
   [root@cass01 ~]# cat /etc/hosts
   ```

   Output

   ```
   127.0.0.1 localhost localhost.localdomain localhost4
   localhost4.localdomain4
   ```

```
::1 localhost localhost.localdomain localhost6
localhost6.localdomain6

## analytics vm
172.20.252.203 host-node
172.20.252.227 cass01.mycorp.com cass01
## end: analytics vm
172.20.252.227 cass01
172.20.252.228 cass02
172.20.252.229 cass03
172.20.252.230 cass04
172.20.252.231 cass05
```

Starting Cassandra

Before starting Cassandra, please be sure to have the correct version of `cassandra.yaml` installed on the nodes. A working copy is included in the appendix see Appendix.

1. Run the following command on all the Cassandra VMs:

```
[root@cass01 ~]# service cassandra start
```

2. Check the status of the Cassandra VMs using:

```
[root@cass01 ~]# nodetool status
```

Example Output

```
Datacenter: dc1
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address         Load       Tokens       Owns (effective)  Host ID                               Rack
UN  172.20.252.227  25.82 GB   30           62.2%             c856c72b-dc91-4483-b90a-61d4d7b22c69  rack1
UN  172.20.252.228  28.02 GB   30           63.5%             bf6542c1-6614-4c2b-92e2-a1071ba69d65  rack1
UN  172.20.252.229  27.88 GB   30           61.0%             abce1629-fa3c-44ca-87d6-a636f4dded1c  rack1
UN  172.20.252.230  25.61 GB   30           61.4%             b7d469f8-82da-4560-bd5d-5cec4b953141  rack1
UN  172.20.252.231  29.63 GB   30           51.9%             427d98ef-d523-477f-be0a-73be55acc914  rack1
```

Cassandra status:

UN: Cassandra VM is up

DN: Cassandra VM is down

If you expect a particular node to be in the cluster and it isn't, you can log into the node, check if Cassandra is running, and check logs in `/var/log/cassandra`.

Verification

You can verify the status of your Cassandra cluster by running `cassandra-stress`. The following is a script that writes 1 million rows to a cluster consisting of 6 Cassandra nodes.

1. To run the script.

```
#!/bin/bash
HOSTS="cass1,cass2,cass3,cass4,cass5,cass6"
cassandra-stress write n=1000000 cl=local_quorum -node $HOSTS
```

2. The output from running the script is shown below. The script was run from a client which was another VM with Cassandra installed on it. The client was not part of the cluster, however it was on the same network.

Output

```
Results:
op rate : 51963 [WRITE:51963]
partition rate : 51963 [WRITE:51963]
row rate : 51963 [WRITE:51963]
latency mean : 3.8 [WRITE:3.8]
latency median : 1.5 [WRITE:1.5]
latency 95th percentile : 12.6 [WRITE:12.6]
latency 99th percentile : 36.2 [WRITE:36.2]
latency 99.9th percentile : 64.5 [WRITE:64.5]
latency max : 131.9 [WRITE:131.9]
Total partitions : 1000000 [WRITE:1000000]
Total errors : 0 [WRITE:0]
total gc count : 0
total gc mb : 0
total gc time (s) : 0
avg gc time(ms) : NaN
stdev gc time(ms) : 0
Total operation time : 00:00:19
```

Appendix

This appendix describes the procedure for installing and using the `cassandra.yaml` file.

- Procedure
- YAML file

Procedure

This appendix contains a working copy of the `cassandra.yaml` for Cassandra 3.0. This file must be copied to:

`/etc/cassandra/conf/cassandra.yaml`

Once the file has finished copying you must restart Cassandra.

Important: you now need to make at least 3 changes in the `cassandra.yaml` file for your specific environment, see below for details.

1. The seeds entry, which tells a joining node to contact this node to learn the topology of the ring. Change:

   `- seeds: "X.X.X.X"`

   to

   `- seeds: "10.1.1.10"`

   Where 10.1.1.10 is the IP address of any one of the Cassandra VMs in your cluster.

Note: the seeds entry (or a list of seed entries) must stay the same for all nodes in your Cassandra cluster.

2. `Rpc_address`: This is the listen address for client connections. Change:

   `rpc_address: XXXX`

   to

   `rpc_address: cass01`

   Where `cass01` is the hostname of the Cassandra VM the YAML file is installed on.

Note: this line must be changed for every VM in your Cassandra cluster.

3. `auto_bootstrap`: This value is set to false during initialization. It must be changed after initialization, as described in the link included in the file.

YAML file

The following is a copy of the cassandra.yaml file:

```
# Cassandra storage config YAML
# Please search for the word "HSP" to detect the parameters that
# need to be changed.
# NOTE:
# See http://wiki.apache.org/cassandra/StorageConfiguration for
# full explanations of configuration directives
# /NOTE
# The name of the cluster. This is mainly used to prevent machines in
# one logical cluster from joining another.
cluster_name: 'My_Cassandra_Cluster'
# This defines the number of tokens randomly assigned to this node on the ring
# The more tokens, relative to other nodes, the larger the proportion of data
# that this node will store. You probably want all nodes to have the same number
# of tokens assuming they have equal hardware capability.
#
# If you leave this unspecified, Cassandra will use the default of 1 token for legacy
compatibility,
# and will use the initial_token as described below.
#
# Specifying initial_token will override this setting on the node's initial start,
# on subsequent starts, this setting will apply even if initial token is set.
#
# If you already have a cluster with 1 token per node, and wish to migrate to
# multiple tokens per node, see http://wiki.apache.org/cassandra/Operations
num_tokens: 30
# Triggers automatic allocation of num_tokens tokens for this node. The allocation
# algorithm attempts to choose tokens in a way that optimizes replicated load over
# the nodes in the datacenter for the replication strategy used by the specified
# keyspace.
#
# The load assigned to each node will be close to proportional to its number of
# vnodes.
#
# Only supported with the Murmur3Partitioner.
# allocate_tokens_for_keyspace: KEYSPACE
# initial_token allows you to specify tokens manually. While you can use # it with
# vnodes (num_tokens > 1, above) -- in which case you should provide a
# comma-separated list -- it's primarily used when adding nodes # to legacy clusters
# that do not have vnodes enabled.
# initial_token:
# See http://wiki.apache.org/cassandra/HintedHandoff
# May either be "true" or "false" to enable globally
hinted_handoff_enabled: true
# When hinted_handoff_enabled is true, a black list of data centers that will not
# perform hinted handoff
#hinted_handoff_disabled_datacenters:
```

```
# - DC1
# - DC2
# this defines the maximum amount of time a dead host will have hints
# generated. After it has been dead this long, new hints for it will not be
# created until it has been seen alive and gone down again.
max_hint_window_in_ms: 10800000 # 3 hours
# Maximum throttle in KBs per second, per delivery thread. This will be
# reduced proportionally to the number of nodes in the cluster. (If there
# are two nodes in the cluster, each delivery thread will use the maximum
# rate; if there are three, each will throttle to half of the maximum,
# since we expect two nodes to be delivering hints simultaneously.)
hinted_handoff_throttle_in_kb: 1024
# Number of threads with which to deliver hints;
# Consider increasing this number when you have multi-dc deployments, since
# cross-dc handoff tends to be slower
max_hints_delivery_threads: 2
# Directory where Cassandra should store hints.
# If not set, the default directory is $CASSANDRA_HOME/data/hints.
hints_directory: /var/lib/cassandra/hints
# How often hints should be flushed from the internal buffers to disk.
# Will *not* trigger fsync.
hints_flush_period_in_ms: 10000
# Maximum size for a single hints file, in megabytes.
max_hints_file_size_in_mb: 128
# Compression to apply to the hint files. If omitted, hints files
# will be written uncompressed. LZ4, Snappy, and Deflate compressors
# are supported.
#hints_compression:
# - class_name: LZ4Compressor
# parameters:
# -
# Maximum throttle in KBs per second, total. This will be
# reduced proportionally to the number of nodes in the cluster.
batchlog_replay_throttle_in_kb: 1024
# Authentication backend, implementing IAuthenticator; used to identify users
# Out of the box, Cassandra provides org.apache.cassandra.auth.
{AllowAllAuthenticator,
# PasswordAuthenticator}.
#
# - AllowAllAuthenticator performs no checks - set it to disable authentication.
# - PasswordAuthenticator relies on username/password pairs to authenticate
# users. It keeps usernames and hashed passwords in system_auth.credentials table.
# Please increase system_auth keyspace replication factor if you use this
authenticator.
# If using PasswordAuthenticator, CassandraRoleManager must also be used (see below)
authenticator: AllowAllAuthenticator
# Authorization backend, implementing IAuthorizer; used to limit access/provide
permissions
# Out of the box, Cassandra provides org.apache.cassandra.auth.{AllowAllAuthorizer,
# CassandraAuthorizer}.
#
# - AllowAllAuthorizer allows any action to any user - set it to disable
authorization.
```

```
# - CassandraAuthorizer stores permissions in system_auth.permissions table. Please
# increase system_auth keyspace replication factor if you use this authorizer.
authorizer: AllowAllAuthorizer
# Part of the Authentication & Authorization backend, implementing IRoleManager; used
# to maintain grants and memberships between roles.
# Out of the box, Cassandra provides org.apache.cassandra.auth.CassandraRoleManager,
# which stores role information in the system_auth keyspace. Most functions of the
# IRoleManager require an authenticated login, so unless the configured
IAuthenticator
# actually implements authentication, most of this functionality will be unavailable.
#
# - CassandraRoleManager stores role data in the system_auth keyspace. Please
# increase system_auth keyspace replication factor if you use this role manager.
role_manager: CassandraRoleManager
# Validity period for roles cache (fetching permissions can be an
# expensive operation depending on the authorizer). Granted roles are cached for
# authenticated sessions in AuthenticatedUser and after the period specified
# here, become eligible for (async) reload.
# Defaults to 2000, set to 0 to disable.
# Will be disabled automatically for AllowAllAuthenticator.
roles_validity_in_ms: 2000
# Refresh interval for roles cache (if enabled).
# After this interval, cache entries become eligible for refresh. Upon next
# access, an async reload is scheduled and the old value returned until it
# completes. If roles_validity_in_ms is non-zero, then this must be
# also.
# Defaults to the same value as roles_validity_in_ms.
# roles_update_interval_in_ms: 1000
# Validity period for permissions cache (fetching permissions can be an
# expensive operation depending on the authorizer, CassandraAuthorizer is
# one example). Defaults to 2000, set to 0 to disable.
# Will be disabled automatically for AllowAllAuthorizer.
permissions_validity_in_ms: 2000
# Refresh interval for permissions cache (if enabled).
# After this interval, cache entries become eligible for refresh. Upon next
# access, an async reload is scheduled and the old value returned until it
# completes. If permissions_validity_in_ms is non-zero, then this must be
# also.
# Defaults to the same value as permissions_validity_in_ms.
# permissions_update_interval_in_ms: 1000
# The partitioner is responsible for distributing groups of rows (by
# partition key) across nodes in the cluster. You should leave this
# alone for new clusters. The partitioner can NOT be changed without
# reloading all data, so when upgrading you should set this to the
# same partitioner you were already using.
#
# Besides Murmur3Partitioner, partitioners included for backwards
# compatibility include RandomPartitioner, ByteOrderedPartitioner, and
# OrderPreservingPartitioner.
#
partitioner: org.apache.cassandra.dht.Murmur3Partitioner
# Directories where Cassandra should store data on disk. Cassandra
# will spread data evenly across them, subject to the granularity of
```

```
# the configured compaction strategy.
# If not set, the default directory is $CASSANDRA_HOME/data/data.
data_file_directories:
- /var/lib/cassandra/data
# commit log. when running on magnetic HDD, this should be a
# separate spindle than the data directories.
# If not set, the default directory is $CASSANDRA_HOME/data/commitlog.
commitlog_directory: /var/lib/cassandra/commitlog
# policy for data disk failures:
# die: shut down gossip and client transports and kill the JVM for any fs errors or
# single-sstable errors, so the node can be replaced.
# stop_paranoid: shut down gossip and client transports even for single-sstable
errors,
# kill the JVM for errors during startup.
# stop: shut down gossip and client transports, leaving the node effectively dead,
but
# can still be inspected via JMX, kill the JVM for errors during startup.
# best_effort: stop using the failed disk and respond to requests based on
# remaining available sstables. This means you WILL see obsolete
# data at CL.ONE!
# ignore: ignore fatal errors and let requests fail, as in pre-1.2 Cassandra
disk_failure_policy: stop
# policy for commit disk failures:
# die: shut down gossip and Thrift and kill the JVM, so the node can be replaced.
# stop: shut down gossip and Thrift, leaving the node effectively dead, but
# can still be inspected via JMX.
# stop_commit: shutdown the commit log, letting writes collect but
# continuing to service reads, as in pre-2.0.5 Cassandra
# ignore: ignore fatal errors and let the batches fail
commit_failure_policy: stop
# Maximum size of the key cache in memory.
#
# Each key cache hit saves 1 seek and each row cache hit saves 2 seeks at the
# minimum, sometimes more. The key cache is fairly tiny for the amount of
# time it saves, so it's worthwhile to use it at large numbers.
# The row cache saves even more time, but must contain the entire row,
# so it is extremely space-intensive. It's best to only use the
# row cache if you have hot rows or static rows.
#
# NOTE: if you reduce the size, you may not get you hottest keys loaded on startup.
#
# Default value is empty to make it "auto" (min(5% of Heap (in MB), 100MB)). Set to 0
to disable key cache.
key_cache_size_in_mb:
# Duration in seconds after which Cassandra should
# save the key cache. Caches are saved to saved_caches_directory as
# specified in this configuration file.
#
# Saved caches greatly improve cold-start speeds, and is relatively cheap in
# terms of I/O for the key cache. Row cache saving is much more expensive and
# has limited use.
#
# Default is 14400 or 4 hours.
```

```
key_cache_save_period: 14400
# Number of keys from the key cache to save
# Disabled by default, meaning all keys are going to be saved
# key_cache_keys_to_save: 100
# Row cache implementation class name.
# Available implementations:
# org.apache.cassandra.cache.OHCProvider Fully off-heap row cache implementation
(default).
# org.apache.cassandra.cache.SerializingCacheProvider This is the row cache
implementation availabile
# in previous releases of Cassandra.
# row_cache_class_name: org.apache.cassandra.cache.OHCProvider
# Maximum size of the row cache in memory.
# Please note that OHC cache implementation requires some additional off-heap memory
to manage
# the map structures and some in-flight memory during operations before/after cache
entries can be
# accounted against the cache capacity. This overhead is usually small compared to
the whole capacity.
# Do not specify more memory that the system can afford in the worst usual situation
and leave some
# headroom for OS block level cache. Do never allow your system to swap.
#
# Default value is 0, to disable row caching.
row_cache_size_in_mb: 0
# Duration in seconds after which Cassandra should save the row cache.
# Caches are saved to saved_caches_directory as specified in this configuration file.
#
# Saved caches greatly improve cold-start speeds, and is relatively cheap in
# terms of I/O for the key cache. Row cache saving is much more expensive and
# has limited use.
#
# Default is 0 to disable saving the row cache.
row_cache_save_period: 0
# Number of keys from the row cache to save.
# Specify 0 (which is the default), meaning all keys are going to be saved
# row_cache_keys_to_save: 100
# Maximum size of the counter cache in memory.
#
# Counter cache helps to reduce counter locks' contention for hot counter cells.
# In case of RF = 1 a counter cache hit will cause Cassandra to skip the read before
# write entirely. With RF > 1 a counter cache hit will still help to reduce the
duration
# of the lock hold, helping with hot counter cell updates, but will not allow
skipping
# the read entirely. Only the local (clock, count) tuple of a counter cell is kept
# in memory, not the whole counter, so it's relatively cheap.
#
# NOTE: if you reduce the size, you may not get you hottest keys loaded on startup.
#
# Default value is empty to make it "auto" (min(2.5% of Heap (in MB), 50MB)). Set to
0 to disable counter cache.
```

```
# NOTE: if you perform counter deletes and rely on low gcgs, you should disable the
counter cache.
counter_cache_size_in_mb:
# Duration in seconds after which Cassandra should
# save the counter cache (keys only). Caches are saved to saved_caches_directory as
# specified in this configuration file.
#
# Default is 7200 or 2 hours.
counter_cache_save_period: 7200
# Number of keys from the counter cache to save
# Disabled by default, meaning all keys are going to be saved
# counter_cache_keys_to_save: 100
# saved caches
# If not set, the default directory is $CASSANDRA_HOME/data/saved_caches.
saved_caches_directory: /var/lib/cassandra/saved_caches
# commitlog_sync may be either "periodic" or "batch."
#
# When in batch mode, Cassandra won't ack writes until the commit log
# has been fsynced to disk. It will wait
# commitlog_sync_batch_window_in_ms milliseconds between fsyncs.
# This window should be kept short because the writer threads will
# be unable to do extra work while waiting. (You may need to increase
# concurrent_writes for the same reason.)
#
# commitlog_sync: batch
# commitlog_sync_batch_window_in_ms: 2
#
# the other option is "periodic" where writes may be acked immediately
# and the CommitLog is simply synced every commitlog_sync_period_in_ms
# milliseconds.
commitlog_sync: periodic
commitlog_sync_period_in_ms: 10000
# The size of the individual commitlog file segments. A commitlog
# segment may be archived, deleted, or recycled once all the data
# in it (potentially from each columnfamily in the system) has been
# flushed to sstables.
#
# The default size is 32, which is almost always fine, but if you are
# archiving commitlog segments (see commitlog_archiving.properties),
# then you probably want a finer granularity of archiving; 8 or 16 MB
# is reasonable.
# Max mutation size is also configurable via max_mutation_size_in_kb setting in
# cassandra.yaml. The default is half the size commitlog_segment_size_in_mb * 1024.
#
# NOTE: If max_mutation_size_in_kb is set explicitly then commitlog_segment_size_in_
mb must
# be set to at least twice the size of max_mutation_size_in_kb / 1024
#
commitlog_segment_size_in_mb: 32
# Compression to apply to the commit log. If omitted, the commit log
# will be written uncompressed. LZ4, Snappy, and Deflate compressors
# are supported.
#commitlog_compression:
```

```
# - class_name: LZ4Compressor
# parameters:
# -
# any class that implements the SeedProvider interface and has a
# constructor that takes a Map<String, String> of parameters will do.
seed_provider:
# Addresses of hosts that are deemed contact points.
# Cassandra nodes use this list of hosts to find each other and learn
# the topology of the ring. You must change this if you are running
# multiple nodes!
- class_name: org.apache.cassandra.locator.SimpleSeedProvider
parameters:
# seeds is actually a comma-delimited list of addresses.
# Ex: "<ip1>,<ip2>,<ip3>"
# Dear HSP user, please change this IP to the IP of any node in your Cassandra
cluster.
- seeds: "X.X.X.X"
# For workloads with more data than can fit in memory, Cassandra's
# bottleneck will be reads that need to fetch data from
# disk. "concurrent_reads" should be set to (16 * number_of_drives) in
# order to allow the operations to enqueue low enough in the stack
# that the OS and drives can reorder them. Same applies to
# "concurrent_counter_writes", since counter writes read the current
# values before incrementing and writing them back.
#
# On the other hand, since writes are almost never IO bound, the ideal
# number of "concurrent_writes" is dependent on the number of cores in
# your system; (8 * number_of_cores) is a good rule of thumb.
concurrent_reads: 128
concurrent_writes: 128
concurrent_counter_writes: 128
# For materialized view writes, as there is a read involved, so this should
# be limited by the less of concurrent reads or concurrent writes.
concurrent_materialized_view_writes: 32
# Maximum memory to use for pooling sstable buffers. Defaults to the smaller
# of 1/4 of heap or 512MB. This pool is allocated off-heap, so is in addition
# to the memory allocated for heap. Memory is only allocated as needed.
# file_cache_size_in_mb: 512
# Flag indicating whether to allocate on or off heap when the sstable buffer
# pool is exhausted, that is when it has exceeded the maximum memory
# file_cache_size_in_mb, beyond which it will not cache buffers but allocate on
request.
# buffer_pool_use_heap_if_exhausted: true
# The strategy for optimizing disk read
# Possible values are:
# ssd (for solid state disks, the default)
# spinning (for spinning disks)
disk_optimization_strategy: ssd
# Total permitted memory to use for memtables. Cassandra will stop
# accepting writes when the limit is exceeded until a flush completes,
# and will trigger a flush based on memtable_cleanup_threshold
# If omitted, Cassandra will set both to 1/4 the size of the heap.
# memtable_heap_space_in_mb: 2048
```

```
# memtable_offheap_space_in_mb: 2048
# Ratio of occupied non-flushing memtable size to total permitted size
# that will trigger a flush of the largest memtable. Larger mct will
# mean larger flushes and hence less compaction, but also less concurrent
# flush activity which can make it difficult to keep your disks fed
# under heavy write load.
#
# memtable_cleanup_threshold defaults to 1 / (memtable_flush_writers + 1)
# memtable_cleanup_threshold: 0.11
# Specify the way Cassandra allocates and manages memtable memory.
# Options are:
# heap_buffers: on heap nio buffers
# offheap_buffers: off heap (direct) nio buffers
memtable_allocation_type: heap_buffers
# Total space to use for commit logs on disk.
#
# If space gets above this value, Cassandra will flush every dirty CF
# in the oldest segment and remove it. So a small total commitlog space
# will tend to cause more flush activity on less-active columnfamilies.
#
# The default value is the smaller of 8192, and 1/4 of the total space
# of the commitlog volume.
#
# commitlog_total_space_in_mb: 8192
# This sets the amount of memtable flush writer threads. These will
# be blocked by disk io, and each one will hold a memtable in memory
# while blocked.
#
# memtable_flush_writers defaults to the smaller of (number of disks,
# number of cores), with a minimum of 2 and a maximum of 8.
#
# If your data directories are backed by SSD, you should increase this
# to the number of cores.
#memtable_flush_writers: 8
# A fixed memory pool size in MB for for SSTable index summaries. If left
# empty, this will default to 5% of the heap size. If the memory usage of
# all index summaries exceeds this limit, SSTables with low read rates will
# shrink their index summaries in order to meet this limit. However, this
# is a best-effort process. In extreme conditions Cassandra may need to use
# more than this amount of memory.
index_summary_capacity_in_mb:
# How frequently index summaries should be resampled. This is done
# periodically to redistribute memory from the fixed-size pool to sstables
# proportional their recent read rates. Setting to -1 will disable this
# process, leaving existing index summaries at their current sampling level.
index_summary_resize_interval_in_minutes: 60
# Whether to, when doing sequential writing, fsync() at intervals in
# order to force the operating system to flush the dirty
# buffers. Enable this to avoid sudden dirty buffer flushing from
# impacting read latencies. Almost always a good idea on SSDs; not
# necessarily on platters.
trickle_fsync: true
trickle_fsync_interval_in_kb: 10240
```

```
# TCP port, for commands and data
# For security reasons, you should not expose this port to the internet. Firewall it
if needed.
storage_port: 7000
# SSL port, for encrypted communication. Unused unless enabled in
# encryption_options
# For security reasons, you should not expose this port to the internet. Firewall it
if needed.
ssl_storage_port: 7001
# Address or interface to bind to and tell other Cassandra nodes to connect to.
# You _must_ change this if you want multiple nodes to be able to communicate!
#
# Set listen_address OR listen_interface, not both. Interfaces must correspond
# to a single address, IP aliasing is not supported.
#
# Leaving it blank leaves it up to InetAddress.getLocalHost(). This
# will always do the Right Thing _if_ the node is properly configured
# (hostname, name resolution, etc), and the Right Thing is to use the
# address associated with the hostname (it might not be).
#
# Setting listen_address to 0.0.0.0 is always wrong.
#
# If you choose to specify the interface by name and the interface has an ipv4 and an
ipv6 address
# you can specify which should be chosen using listen_interface_prefer_ipv6. If false
the first ipv4
# address will be used. If true the first ipv6 address will be used. Defaults to
false preferring
# ipv4. If there is only one address it will be selected regardless of ipv4/ipv6.
listen_address:
# listen_interface: eth0
# listen_interface_prefer_ipv6: false
# Address to broadcast to other Cassandra nodes
# Leaving this blank will set it to the same value as listen_address
# broadcast_address: 1.2.3.4
# When using multiple physical network interfaces, set this
# to true to listen on broadcast_address in addition to
# the listen_address, allowing nodes to communicate in both
# interfaces.
# Ignore this property if the network configuration automatically
# routes between the public and private networks such as EC2.
# listen_on_broadcast_address: false
# Internode authentication backend, implementing IInternodeAuthenticator;
# used to allow/disallow connections from peer nodes.
# internode_authenticator: org.apache.cassandra.auth.AllowAllInternodeAuthenticator
# Whether to start the native transport server.
# Please note that the address on which the native transport is bound is the
# same as the rpc_address. The port however is different and specified below.
start_native_transport: true
# port for the CQL native transport to listen for clients on
# For security reasons, you should not expose this port to the internet. Firewall it
if needed.
```

```
native_transport_port: 9042
# Enabling native transport encryption in client_encryption_options allows you to
either use
# encryption for the standard port or to use a dedicated, additional port along with
the unencrypted
# standard native_transport_port.
# Enabling client encryption and keeping native_transport_port_ssl disabled will use
encryption
# for native_transport_port. Setting native_transport_port_ssl to a different value
# from native_transport_port will use encryption for native_transport_port_ssl while
# keeping native_transport_port unencrypted.
# native_transport_port_ssl: 9142
# The maximum threads for handling requests when the native transport is used.
# This is similar to rpc_max_threads though the default differs slightly (and
# there is no native_transport_min_threads, idle threads will always be stopped
# after 30 seconds).
# native_transport_max_threads: 128
#
# The maximum size of allowed frame. Frame (requests) larger than this will
# be rejected as invalid. The default is 256MB. If you're changing this parameter,
# you may want to adjust max_value_size_in_mb accordingly.
# native_transport_max_frame_size_in_mb: 256
# The maximum number of concurrent client connections.
# The default is -1, which means unlimited.
# native_transport_max_concurrent_connections: -1
# The maximum number of concurrent client connections per source ip.
# The default is -1, which means unlimited.
# native_transport_max_concurrent_connections_per_ip: -1
# Whether to start the thrift rpc server.
start_rpc: true
# The address or interface to bind the Thrift RPC service and native transport
# server to.
#
# Set rpc_address OR rpc_interface, not both. Interfaces must correspond
# to a single address, IP aliasing is not supported.
#
# Leaving rpc_address blank has the same effect as on listen_address
# (i.e. it will be based on the configured hostname of the node).
#
# Note that unlike listen_address, you can specify 0.0.0.0, but you must also
# set broadcast_rpc_address to a value other than 0.0.0.0.
#
# For security reasons, you should not expose this port to the internet. Firewall it
if needed.
#
# If you choose to specify the interface by name and the interface has an ipv4 and an
ipv6 address
# you can specify which should be chosen using rpc_interface_prefer_ipv6. If false
the first ipv4
# address will be used. If true the first ipv6 address will be used. Defaults to
false preferring
# ipv4. If there is only one address it will be selected regardless of ipv4/ipv6.
```

```
# Dear HSP user, please change this entry to the hostname or IP of the Cassandra VM
that this
# yaml file is installed on.
# Examples:
# rpc_address: cass1
# rpc_address: 172.20.252.223
rpc_address: XXXX
# rpc_interface: eth1
# rpc_interface_prefer_ipv6: false
# port for Thrift to listen for clients on
rpc_port: 9160
# RPC address to broadcast to drivers and other Cassandra nodes. This cannot
# be set to 0.0.0.0. If left blank, this will be set to the value of
# rpc_address. If rpc_address is set to 0.0.0.0, broadcast_rpc_address must
# be set.
# broadcast_rpc_address: 1.2.3.4
# enable or disable keepalive on rpc/native connections
rpc_keepalive: true
# Cassandra provides two out-of-the-box options for the RPC Server:
#
# sync -> One thread per thrift connection. For a very large number of clients,
memory
# will be your limiting factor. On a 64 bit JVM, 180KB is the minimum stack size
# per thread, and that will correspond to your use of virtual memory (but physical
memory
# may be limited depending on use of stack space).
#
# hsha -> Stands for "half synchronous, half asynchronous." All thrift clients are
handled
# asynchronously using a small number of threads that does not vary with the amount
# of thrift clients (and thus scales well to many clients). The rpc requests are
still
# synchronous (one thread per active request). If hsha is selected then it is
essential
# that rpc_max_threads is changed from the default value of unlimited.
#
# The default is sync because on Windows hsha is about 30% slower. On Linux,
# sync/hsha performance is about the same, with hsha of course using less memory.
#
# Alternatively, can provide your own RPC server by providing the fully-qualified
class name
# of an o.a.c.t.TServerFactory that can create an instance of it.
rpc_server_type: sync
# Uncomment rpc_min|max_thread to set request pool size limits.
#
# Regardless of your choice of RPC server (see above), the number of maximum requests
in the
# RPC thread pool dictates how many concurrent requests are possible (but if you are
using the sync
# RPC server, it also dictates the number of clients that can be connected at all).
#
```

```
# The default is unlimited and thus provides no protection against clients
overwhelming the server. You are
# encouraged to set a maximum that makes sense for you in production, but do keep in
mind that
# rpc_max_threads represents the maximum number of client requests this server may
execute concurrently.
#
# rpc_min_threads: 16
rpc_max_threads: 2048
# uncomment to set socket buffer sizes on rpc connections
# rpc_send_buff_size_in_bytes:
# rpc_recv_buff_size_in_bytes:
# Uncomment to set socket buffer size for internode communication
# Note that when setting this, the buffer size is limited by net.core.wmem_max
# and when not setting it it is defined by net.ipv4.tcp_wmem
# See:
# /proc/sys/net/core/wmem_max
# /proc/sys/net/core/rmem_max
# /proc/sys/net/ipv4/tcp_wmem
# /proc/sys/net/ipv4/tcp_wmem
# and: man tcp
# internode_send_buff_size_in_bytes:
# internode_recv_buff_size_in_bytes:
# Frame size for thrift (maximum message length).
thrift_framed_transport_size_in_mb: 15
# Set to true to have Cassandra create a hard link to each sstable
# flushed or streamed locally in a backups/ subdirectory of the
# keyspace data. Removing these links is the operator's
# responsibility.
incremental_backups: false
# Whether or not to take a snapshot before each compaction. Be
# careful using this option, since Cassandra won't clean up the
# snapshots for you. Mostly useful if you're paranoid when there
# is a data format change.
snapshot_before_compaction: false
# Whether or not a snapshot is taken of the data before keyspace truncation
# or dropping of column families. The STRONGLY advised default of true
# should be used to provide data safety. If you set this flag to false, you will
# lose data on truncation or drop.
auto_snapshot: true
# When executing a scan, within or across a partition, we need to keep the
# tombstones seen in memory so we can return them to the coordinator, which
# will use them to make sure other replicas also know about the deleted rows.
# With workloads that generate a lot of tombstones, this can cause performance
# problems and even exaust the server heap.
# (http://www.datastax.com/dev/blog/cassandra-anti-patterns-queues-and-queue-like-
datasets)
# Adjust the thresholds here if you understand the dangers and want to
# scan more tombstones anyway. These thresholds may also be adjusted at runtime
# using the StorageService mbean.
tombstone_warn_threshold: 1000
tombstone_failure_threshold: 100000
# Granularity of the collation index of rows within a partition.
```

```
# Increase if your rows are large, or if you have a very large
# number of rows per partition. The competing goals are these:
# 1) a smaller granularity means more index entries are generated
# and looking up rows withing the partition by collation column
# is faster
# 2) but, Cassandra will keep the collation index in memory for hot
# rows (as part of the key cache), so a larger granularity means
# you can cache more hot rows
column_index_size_in_kb: 64
# Log WARN on any batch size exceeding this value. 5kb per batch by default.
# Caution should be taken on increasing the size of this threshold as it can lead to
node instability.
batch_size_warn_threshold_in_kb: 64
# Fail any batch exceeding this value. 50kb (10x warn threshold) by default.
batch_size_fail_threshold_in_kb: 640
# Log WARN on any batches not of type LOGGED than span across more partitions than
this limit
unlogged_batch_across_partitions_warn_threshold: 10
# Number of simultaneous compactions to allow, NOT including
# validation "compactions" for anti-entropy repair. Simultaneous
# compactions can help preserve read performance in a mixed read/write
# workload, by mitigating the tendency of small sstables to accumulate
# during a single long running compactions. The default is usually
# fine and if you experience problems with compaction running too
# slowly or too fast, you should look at
# compaction_throughput_mb_per_sec first.
#
# concurrent_compactors defaults to the smaller of (number of disks,
# number of cores), with a minimum of 2 and a maximum of 8.
#
# If your data directories are backed by SSD, you should increase this
# to the number of cores.
#concurrent_compactors: 1
# Throttles compaction to the given total throughput across the entire
# system. The faster you insert data, the faster you need to compact in
# order to keep the sstable count down, but in general, setting this to
# 16 to 32 times the rate you are inserting data is more than sufficient.
# Setting this to 0 disables throttling. Note that this account for all types
# of compaction, including validation compaction.
compaction_throughput_mb_per_sec: 64
# Log a warning when compacting partitions larger than this value
compaction_large_partition_warning_threshold_mb: 100
# When compacting, the replacement sstable(s) can be opened before they
# are completely written, and used in place of the prior sstables for
# any range that has been written. This helps to smoothly transfer reads
# between the sstables, reducing page cache churn and keeping hot rows hot
sstable_preemptive_open_interval_in_mb: 50
# Throttles all outbound streaming file transfers on this node to the
# given total throughput in Mbps. This is necessary because Cassandra does
# mostly sequential IO when streaming data during bootstrap or repair, which
# can lead to saturating the network connection and degrading rpc performance.
# When unset, the default is 200 Mbps or 25 MB/s.
# stream_throughput_outbound_megabits_per_sec: 200
```

```
# Throttles all streaming file transfer between the datacenters,
# this setting allows users to throttle inter dc stream throughput in addition
# to throttling all network stream traffic as configured with
# stream_throughput_outbound_megabits_per_sec
# When unset, the default is 200 Mbps or 25 MB/s
# inter_dc_stream_throughput_outbound_megabits_per_sec: 200
# How long the coordinator should wait for read operations to complete
read_request_timeout_in_ms: 50000
# How long the coordinator should wait for seq or index scans to complete
range_request_timeout_in_ms: 100000
# How long the coordinator should wait for writes to complete
write_request_timeout_in_ms: 20000
# How long the coordinator should wait for counter writes to complete
counter_write_request_timeout_in_ms: 50000
# How long a coordinator should continue to retry a CAS operation
# that contends with other proposals for the same row
cas_contention_timeout_in_ms: 10000
# How long the coordinator should wait for truncates to complete
# (This can be much longer, because unless auto_snapshot is disabled
# we need to flush first so we can snapshot before removing the data.)
truncate_request_timeout_in_ms: 600000
# The default timeout for other, miscellaneous operations
request_timeout_in_ms: 100000
# Enable operation timeout information exchange between nodes to accurately
# measure request timeouts. If disabled, replicas will assume that requests
# were forwarded to them instantly by the coordinator, which means that
# under overload conditions we will waste that much extra time processing
# already-timed-out requests.
#
# Warning: before enabling this property make sure to ntp is installed
# and the times are synchronized between the nodes.
cross_node_timeout: true
# Set socket timeout for streaming operation.
# The stream session is failed if no data/ack is received by any of the participants
# within that period, which means this should also be sufficient to stream a large
# sstable or rebuild table indexes.
# Default value is 86400000ms, which means stale streams timeout after 24 hours.
# A value of zero means stream sockets should never time out.
# streaming_socket_timeout_in_ms: 86400000
# phi value that must be reached for a host to be marked down.
# most users should never need to adjust this.
# phi_convict_threshold: 8
# endpoint_snitch -- Set this to a class that implements
# IEndpointSnitch. The snitch has two functions:
# - it teaches Cassandra enough about your network topology to route
# requests efficiently
# - it allows Cassandra to spread replicas around your cluster to avoid
# correlated failures. It does this by grouping machines into
# "datacenters" and "racks." Cassandra will do its best not to have
# more than one replica on the same "rack" (which may not actually
# be a physical location)
#
# IF YOU CHANGE THE SNITCH AFTER DATA IS INSERTED INTO THE CLUSTER,
```

```
# YOU MUST RUN A FULL REPAIR, SINCE THE SNITCH AFFECTS WHERE REPLICAS
# ARE PLACED.
#
# IF THE RACK A REPLICA IS PLACED IN CHANGES AFTER THE REPLICA HAS BEEN
# ADDED TO A RING, THE NODE MUST BE DECOMMISSIONED AND REBOOTSTRAPPED.
#
# Out of the box, Cassandra provides
# - SimpleSnitch:
# Treats Strategy order as proximity. This can improve cache
# locality when disabling read repair. Only appropriate for
# single-datacenter deployments.
# - GossipingPropertyFileSnitch
# This should be your go-to snitch for production use. The rack
# and datacenter for the local node are defined in
# cassandra-rackdc.properties and propagated to other nodes via
# gossip. If cassandra-topology.properties exists, it is used as a
# fallback, allowing migration from the PropertyFileSnitch.
# - PropertyFileSnitch:
# Proximity is determined by rack and data center, which are
# explicitly configured in cassandra-topology.properties.
# - Ec2Snitch:
# Appropriate for EC2 deployments in a single Region. Loads Region
# and Availability Zone information from the EC2 API. The Region is
# treated as the datacenter, and the Availability Zone as the rack.
# Only private IPs are used, so this will not work across multiple
# Regions.
# - Ec2MultiRegionSnitch:
# Uses public IPs as broadcast_address to allow cross-region
# connectivity. (Thus, you should set seed addresses to the public
# IP as well.) You will need to open the storage_port or
# ssl_storage_port on the public IP firewall. (For intra-Region
# traffic, Cassandra will switch to the private IP after
# establishing a connection.)
# - RackInferringSnitch:
# Proximity is determined by rack and data center, which are
# assumed to correspond to the 3rd and 2nd octet of each node's IP
# address, respectively. Unless this happens to match your
# deployment conventions, this is best used as an example of
# writing a custom Snitch class and is provided in that spirit.
#
# You can use a custom Snitch by setting this to the full class name
# of the snitch, which will be assumed to be on your classpath.
endpoint_snitch: GossipingPropertyFileSnitch
# controls how often to perform the more expensive part of host score
# calculation
dynamic_snitch_update_interval_in_ms: 100
# controls how often to reset all host scores, allowing a bad host to
# possibly recover
dynamic_snitch_reset_interval_in_ms: 600000
# if set greater than zero and read_repair_chance is < 1.0, this will allow
# 'pinning' of replicas to hosts in order to increase cache capacity.
# The badness threshold will control how much worse the pinned host has to be
# before the dynamic snitch will prefer other replicas over it. This is
```

```
# expressed as a double which represents a percentage. Thus, a value of
# 0.2 means Cassandra would continue to prefer the static snitch values
# until the pinned host was 20% worse than the fastest.
dynamic_snitch_badness_threshold: 0.1
# request_scheduler -- Set this to a class that implements
# RequestScheduler, which will schedule incoming client requests
# according to the specific policy. This is useful for multi-tenancy
# with a single Cassandra cluster.
# NOTE: This is specifically for requests from the client and does
# not affect inter node communication.
# org.apache.cassandra.scheduler.NoScheduler - No scheduling takes place
# org.apache.cassandra.scheduler.RoundRobinScheduler - Round robin of
# client requests to a node with a separate queue for each
# request_scheduler_id. The scheduler is further customized by
# request_scheduler_options as described below.
request_scheduler: org.apache.cassandra.scheduler.NoScheduler
# Scheduler Options vary based on the type of scheduler
# NoScheduler - Has no options
# RoundRobin
# - throttle_limit -- The throttle_limit is the number of in-flight
# requests per client. Requests beyond
# that limit are queued up until
# running requests can complete.
# The value of 80 here is twice the number of
# concurrent_reads + concurrent_writes.
# - default_weight -- default_weight is optional and allows for
# overriding the default which is 1.
# - weights -- Weights are optional and will default to 1 or the
# overridden default_weight. The weight translates into how
# many requests are handled during each turn of the
# RoundRobin, based on the scheduler id.
#
# request_scheduler_options:
# throttle_limit: 80
# default_weight: 5
# weights:
# Keyspace1: 1
# Keyspace2: 5
# request_scheduler_id -- An identifier based on which to perform
# the request scheduling. Currently the only valid option is keyspace.
# request_scheduler_id: keyspace
# Enable or disable inter-node encryption
# Default settings are TLS v1, RSA 1024-bit keys (it is imperative that
# users generate their own keys) TLS_RSA_WITH_AES_128_CBC_SHA as the cipher
# suite for authentication, key exchange and encryption of the actual data transfers.
# Use the DHE/ECDHE ciphers if running in FIPS 140 compliant mode.
# NOTE: No custom encryption options are enabled at the moment
# The available internode options are : all, none, dc, rack
#
# If set to dc cassandra will encrypt the traffic between the DCs
# If set to rack cassandra will encrypt the traffic between the racks
#
# The passwords used in these options must match the passwords used when generating
```

```
# the keystore and truststore. For instructions on generating these files, see:
#
http://download.oracle.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide
.html#CreateKeystore
#
server_encryption_options:
internode_encryption: none
keystore: conf/.keystore
keystore_password: cassandra
truststore: conf/.truststore
truststore_password: cassandra
# More advanced defaults below:
# protocol: TLS
# algorithm: SunX509
# store_type: JKS
# cipher_suites: [TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA,TLS_DHE_
RSA_WITH_AES_128_CBC_SHA,TLS_DHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_128_
CBC_SHA,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA]
# require_client_auth: false
# enable or disable client/server encryption.
client_encryption_options:
enabled: false
# If enabled and optional is set to true encrypted and unencrypted connections are
handled.
optional: false
keystore: conf/.keystore
keystore_password: cassandra
# require_client_auth: false
# Set trustore and truststore_password if require_client_auth is true
# truststore: conf/.truststore
# truststore_password: cassandra
# More advanced defaults below:
# protocol: TLS
# algorithm: SunX509
# store_type: JKS
# cipher_suites: [TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA,TLS_DHE_
RSA_WITH_AES_128_CBC_SHA,TLS_DHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_128_
CBC_SHA,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA]
# internode_compression controls whether traffic between nodes is
# compressed.
# can be: all - all traffic is compressed
# dc - traffic between different datacenters is compressed
# none - nothing is compressed.
internode_compression: dc
# Enable or disable tcp_nodelay for inter-dc communication.
# Disabling it will result in larger (but fewer) network packets being sent,
# reducing overhead from the TCP protocol itself, at the cost of increasing
# latency if you block for cross-datacenter responses.
inter_dc_tcp_nodelay: false
# TTL for different trace types used during logging of the repair process.
tracetype_query_ttl: 86400
tracetype_repair_ttl: 604800
```

```
# GC Pauses greater than gc_warn_threshold_in_ms will be logged at WARN level
# Adjust the threshold based on your application throughput requirement
# By default, Cassandra logs GC Pauses greater than 200 ms at INFO level
gc_warn_threshold_in_ms: 1000
# UDFs (user defined functions) are disabled by default.
# As of Cassandra 3.0 there is a sandbox in place that should prevent execution of
evil code.
enable_user_defined_functions: false
# Enables scripted UDFs (JavaScript UDFs).
# Java UDFs are always enabled, if enable_user_defined_functions is true.
# Enable this option to be able to use UDFs with "language javascript" or any custom
JSR-223 provider.
# This option has no effect, if enable_user_defined_functions is false.
enable_scripted_user_defined_functions: false
# The default Windows kernel timer and scheduling resolution is 15.6ms for power
conservation.
# Lowering this value on Windows can provide much tighter latency and better
throughput, however
# some virtualized environments may see a negative performance impact from changing
this setting
# below their system default. The sysinternals 'clockres' tool can confirm your
system's default
# setting.
windows_timer_interval: 1
# Maximum size of any value in SSTables. Safety measure to detect SSTable corruption
# early. Any value size larger than this threshold will result into marking an
SSTable
# as corrupted.
# max_value_size_in_mb: 256
# Dear HSP user, please comment this out after the nodes are initialized. Please see:
# https://docs.datastax.com/en/cassandra/2.1/cassandra/configuration/configCassandra_
yaml_r.html?scroll=reference_ds_qfg_n1r_1k__auto_bootstrap
auto_bootstrap: false
```

Hitachi Data Systems

Corporate Headquarters
2845 Lafayette Street
Santa Clara, California 95050-2627
U.S.A.
www.hds.com

Regional Contact Information

Americas
+1 408 970 1000
info@hds.com

Europe, Middle East, and Africa
+44 (0) 1753 618000
info.emea@hds.com

Asia Pacific
+852 3189 7900
hds.marketing.apac@hds.com

Hitachi Data Systems

MK-95HSP027-00